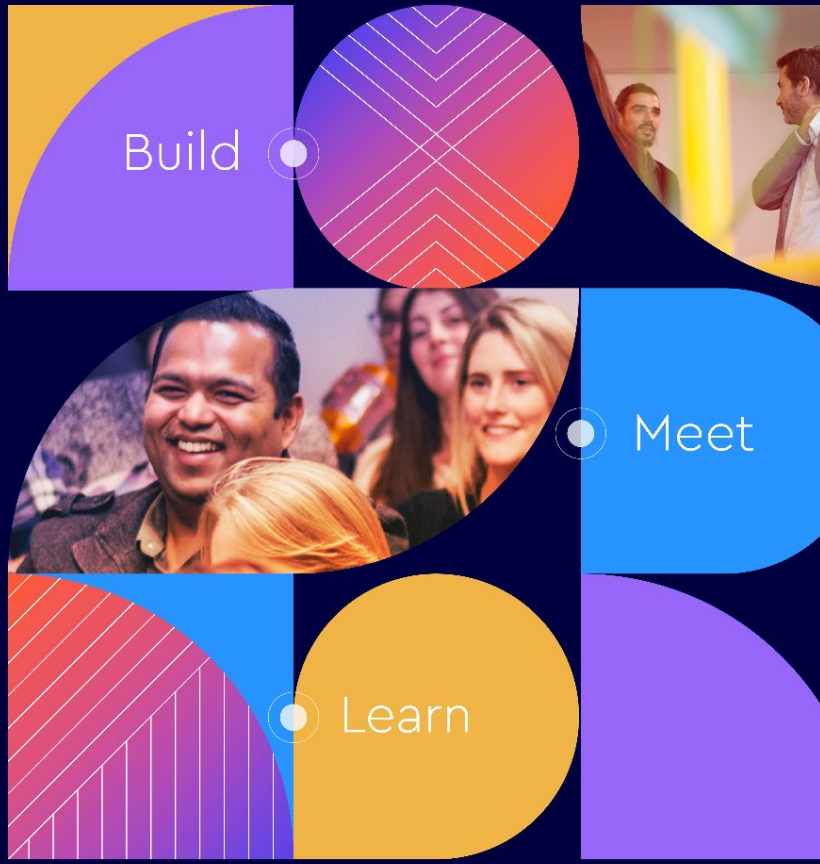


YugabyteDB Voyager

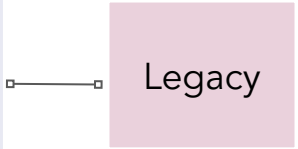
Simplify Data Migration to Accelerate Cloud Adoption

YB Voyager

Motivation

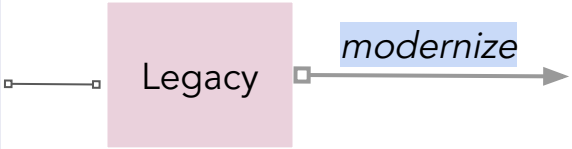
Motivation: Modernize!

- Higher TCO
- Doesn't / Expensive to scale
- Lower performance
- Unable to meet evolving business needs
- +other reasons



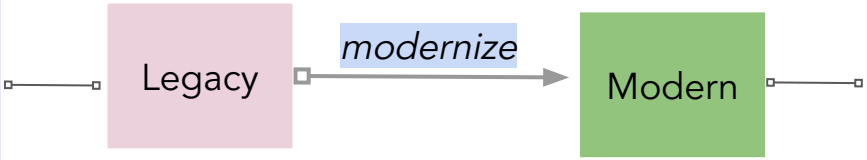
Motivation: Modernize!

- Higher TCO
- Doesn't / Expensive to scale
- Lower performance
- Unable to meet evolving business needs
- +other reasons



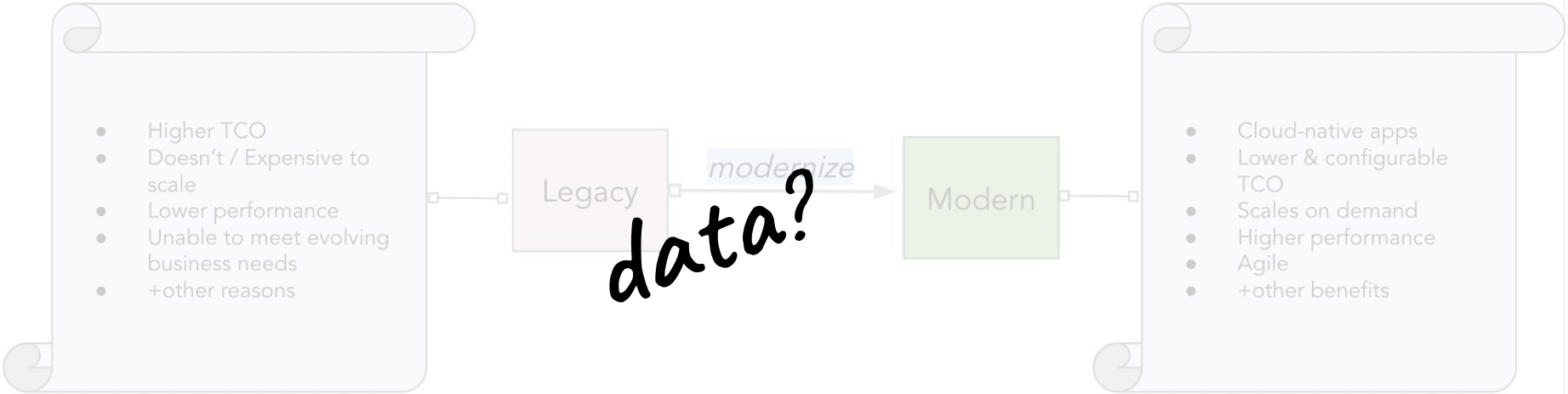
Motivation: Modernize!

- Higher TCO
- Doesn't / Expensive to scale
- Lower performance
- Unable to meet evolving business needs
- +other reasons



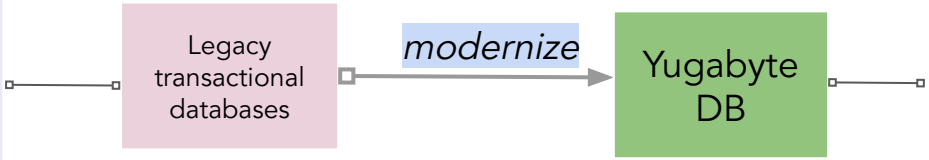
- Cloud-native apps
- Lower & configurable TCO
- Scales on demand
- Higher performance
- Agile
- +other benefits

Motivation: Modernize!



Motivation: Modernize your *data*

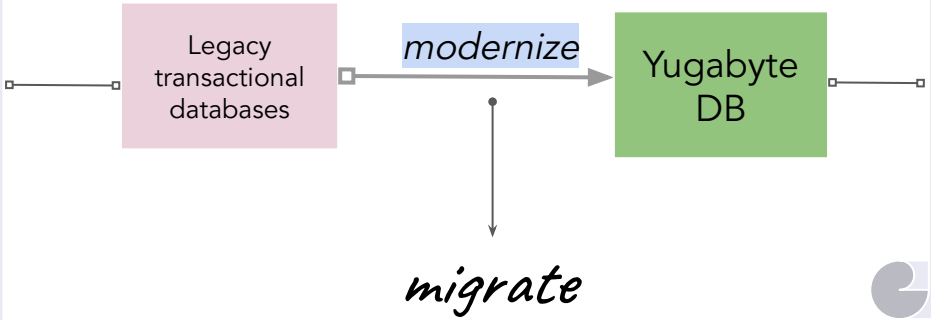
- Expensive licensing model
- Proprietary tech
- Doesn't scale, not *distributed*
- Tied to a cloud provider
- Doesn't meet evolving security and compliance needs
- OSS yet Monolithic
- +other limitations



- Cloud-native database
- Distributed SQL
- Lower TCO
- Scales horizontally & vertically
- Higher performance
- Postgres compatible
- +other benefits

Motivation: Modernize your *data*

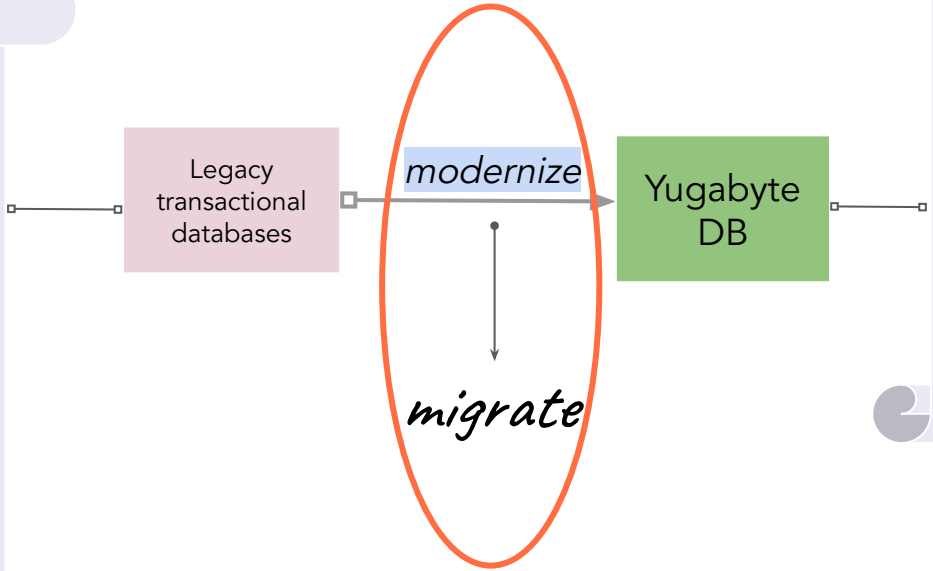
- Expensive licensing model
- Proprietary tech
- Doesn't scale, not *distributed*
- Tied to a cloud provider
- Doesn't meet evolving security and compliance needs
- OSS yet Monolithic
- +other limitations



- Cloud-native database
- Distributed SQL
- Lower TCO
- Scales horizontally & vertically
- Higher performance
- Postgres compatible
- +other benefits

Motivation: *migrate to a CN database to modernize your data*

- Expensive licensing model
- Proprietary tech
- Doesn't scale, not *distributed*
- Tied to a cloud provider
- Doesn't meet evolving security and compliance needs
- OSS yet Monolithic
- +other limitations



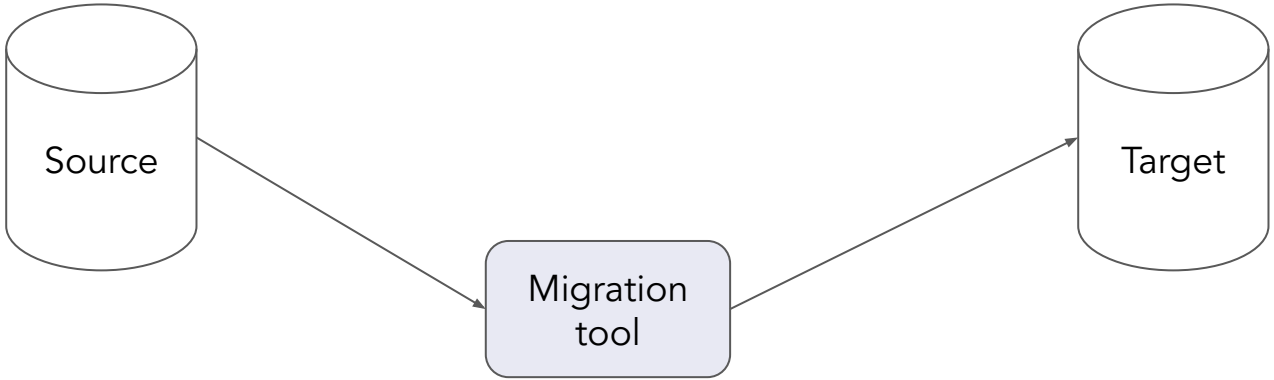
- Cloud-native database
- Distributed SQL
- Lower TCO
- Scales horizontally & vertically
- Higher performance
- Postgres compatible
- +other benefits

Data Migration

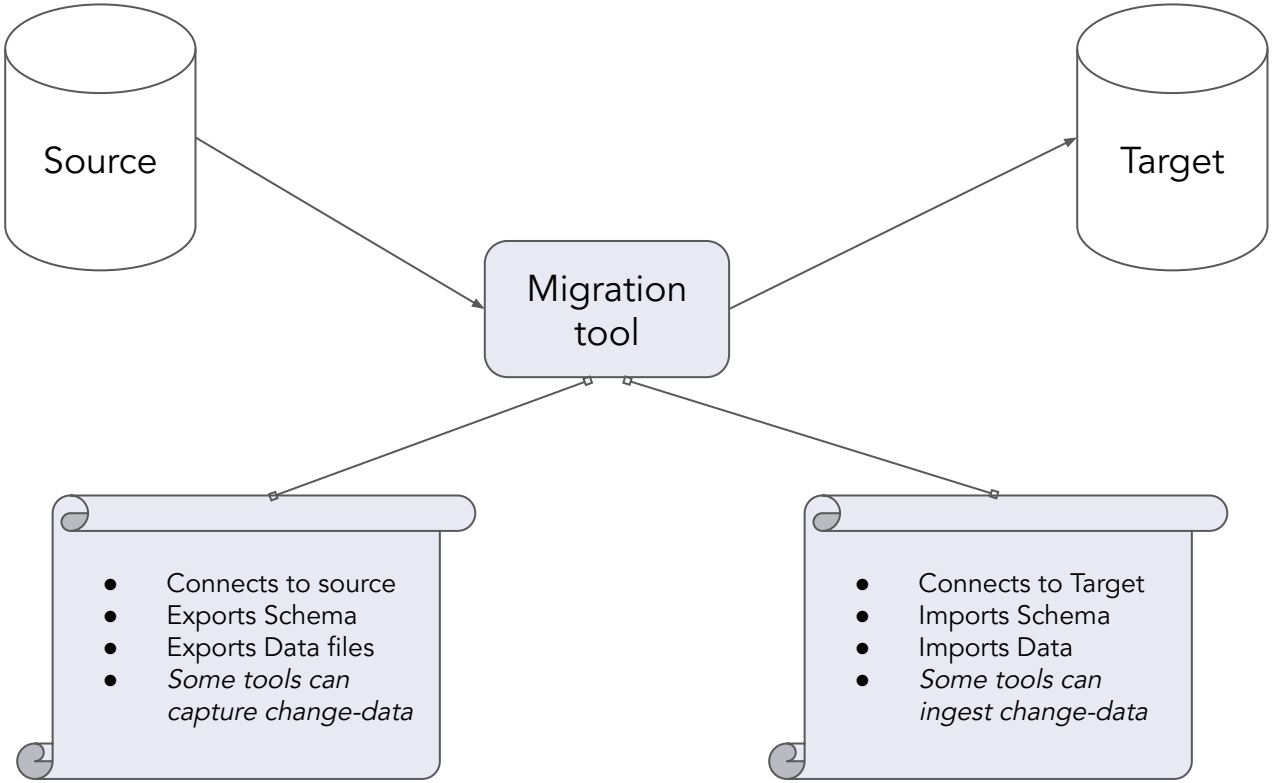
Challenges

Typical Migration process

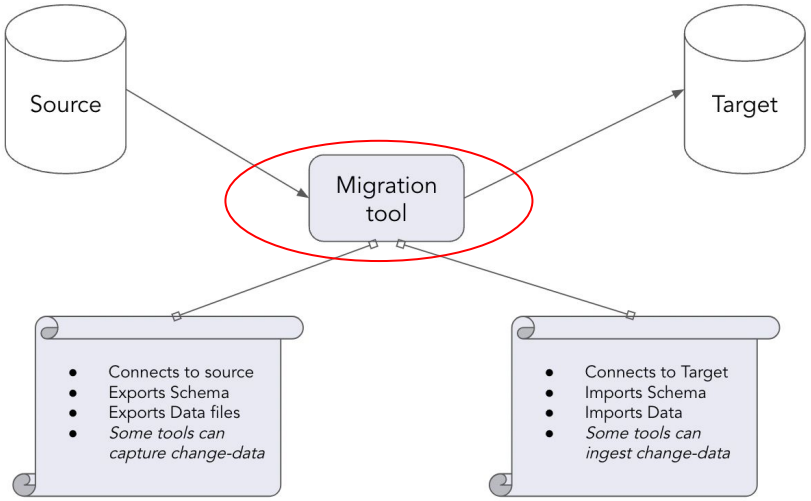
Typical Migration process



Typical Migration process



Typical Migration tools - *atypical* for Distributed SQL



Works just fine when the Target is also a monolithic/single-node database

Unaware of the strengths of a distributed database and doesn't scale.

Some tools depend on computational clusters to prep data sets to be ingested into a distributed database.

YB Voyager

Requirements: when migrating to Distributed SQL

YB Voyager: Goals & Requirements

OFFLINE
migration

ONLINE
migration

FAILBACK

YB Voyager: Goals & Requirements

OFFLINE migration	❑ Export source database schema/objects ❑ Generate a report for users to verify ❑ Create schema objects on YB ❑ Bulk load data ❑ Do validation and generate a final report
ONLINE migration	❑ All from above (OFFLINE mode) ❑ Stream change-data to YB
FAILBACK	❑ Replicate changes from YB to original source DB ❑ Mitigates risk of migrating to cloud

YB Voyager: Goals & Requirements

OFFLINE migration	❑ Export source database schema/objects ❑ Generate a report for users to verify ❑ Create schema objects on YB ❑ Bulk load data ❑ Do validation and generate a final report
ONLINE migration	❑ All from above (OFFLINE mode) ❑ Stream change-data to YB
FAILBACK	❑ Replicate changes from YB to original source DB ❑ Mitigates risk of migrating to cloud

- ➔ User friendly CLI
- ➔ Chunk the source files
- ➔ Parallel data ingestion based on Target YB cluster config
- ➔ Migrate data to YB from flat files & popular databases like Oracle, MySQL etc
- ➔ Same UX/steps for all supported source systems
- ➔ Safe defaults (with override):
  - ◆ #of connections based on target DB config
  - ◆ Ingest batch size
  - ◆ Disable transaction mode
  - ◆ Upsert mode
- ➔ Idempotent & should restart/resume from failed step

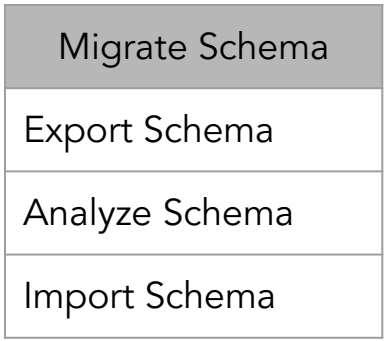
YB Voyager: OFFLINE Migration process flow

1

Prepare	
Install Voyager	Voyager is installed on a VM with sufficient disk space
Prepare source DB	Verify connectivity and right level of access to source DB
Prepare target DB	Verify connectivity and right level of access to target Yugabyte DB

YB Voyager: OFFLINE Migration process flow

2



Use CLI commands to export schema and objects. Tables, Views, SPs, Functions, Triggers, Indexes, Sequence are all (DDLs) exported. SPs/Functions are translated to Postgres compatible syntax

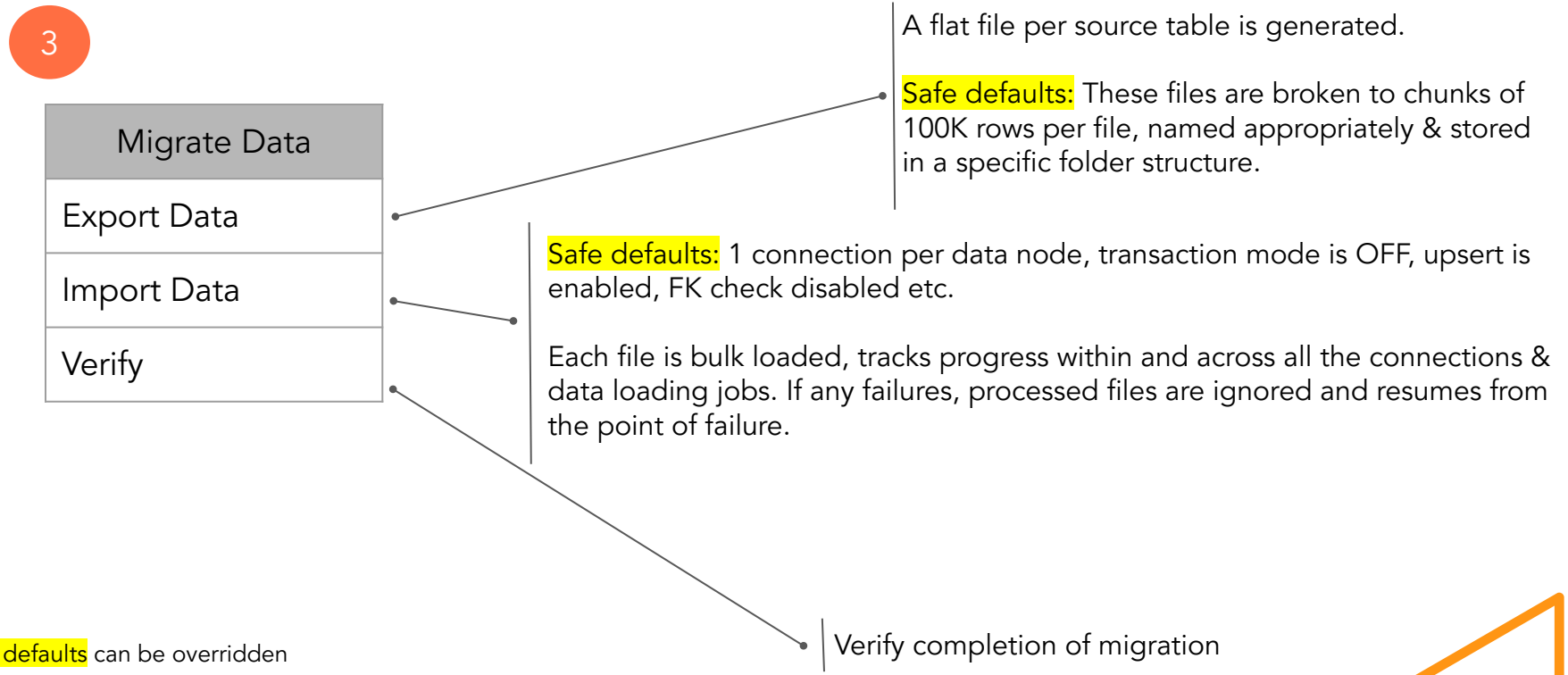
Optional step for the User to manually inspect, verify and make tweaks to the DDLs if necessary

In this step, the appropriate objects are created on the target YB database.

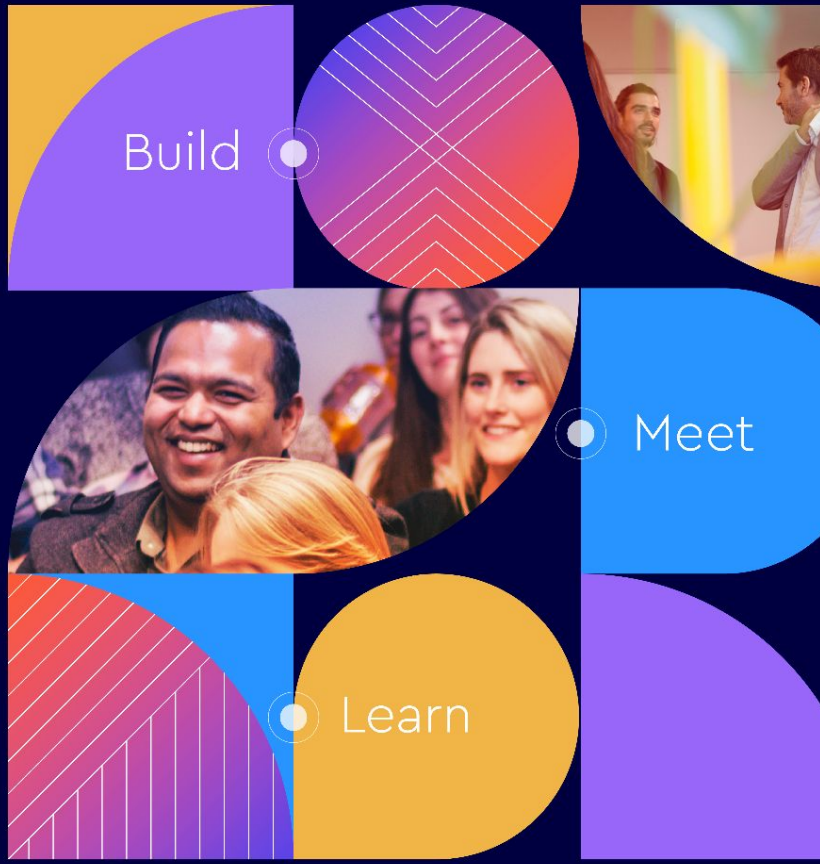
Safe defaults: Indexes are deferred & created only after data load, foreign key check are disabled, triggers are created but disabled

*Safe defaults can be overridden

YB Voyager: OFFLINE Migration process flow



*Safe defaults can be overridden



Thank You

Join us on Slack: yugabyte.com/slack

Star us on Github:
github.com/yugabyte/yugabyte-db